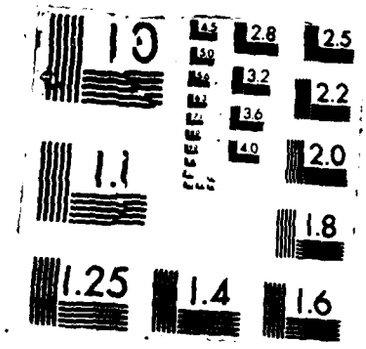
AD-A183 328 OBSERVATIONS ON MULTI-PEG TOWERS OF HANOI(U) ROCHESTER 1/1
UNIV NY DEPT OF COMPUTER SCIENCE R NEHMAN-WOLFE JUL 86
TR-187 ETL-0476 DACA76-85-C-0001

UNCLASSIFIED F/G 12/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

Observations on Multi-Peg Towers of Hanoi

Richard Newman-Wolfe
Department of Computer Science
The University of Rochester
Rochester, NY 14627

TR 187
July 1986

Department of Computer Science
University of Rochester
Rochester, New York 14627

Observations on Multi-Peg Towers of Hanoi

Introduction

While investigating the relative power of abstract storage models, a subproblem that is a generalization of the classic Towers of Hanoi problem arose [Ball '92], [Ball & Coxeter '74]. The standard version has been used to teach recursive methods in computer science [Aho, Hopcroft & Ullman '74], [Wirth '76], and recursion removal [Partsch & Pepper '76]. There have been a number of papers dealing with various recursive and iterative algorithms [Hayes '77], [Buneman & Levy '80], [Walsh '82], all of which have the same sequence of moves [Wood '81], but require differing amounts of program state memory. The classic problem has also been related to the Gray code [Gardner '72], [Buneman & Levy '80], [Er '82].

A generalization of the nature of changing the starting configuration has been investigated ([Er '83]), which is referred to in the literature as the Generalized Towers of Hanoi. In addition, restriction to movement in only one direction (the Cyclic Towers of Hanoi [Atkinson '81], [Walsh '83], [Er '84b] and its generalization of restricting some disks to move in one direction and the rest to move in the other direction (the Colour Towers of Hanoi [Er '84a] and Generalized Colour Towers of Hanoi [Er '84c]) have been studied. The problem with tree-like stacks instead of stacks has been approached [Engelfriet '81], [Jurgenson and Wood '83], and is one natural generalization of the problem. The original problem can be viewed as a unary tree in this context. For the Trees of Hanoi, a number of work pegs equal to the arity of the trees is required. Jurgenson and Wood do ask the question of how having

multiple sets of additional work pegs will affect the solution as an open question at the conclusion of their paper.

However, solutions to the most obvious generalization of the classic problem, that of a different number of pegs, has not appeared in published literature to my knowledge. This generalization was first introduced to me by Pat Hayes in late 1983, at which time Peter Gacs did some interesting upper bounds work, although that was only written up recently [Gacs '86]. Gacs does not not claim originality, rather independence, citing personal communications from Janos Kolmos to the effect that is considered Hungarian folklore, and that Imre Csiszar had done similar work.

The classical Towers of Hanoi game involves three pegs, one of them with a pile of disks of different sizes on it in order, from the smallest at the top to the largest at the bottom. The problem is to move the tower of disks from the peg they are on to another peg, moving one disk at a time from its peg to another peg but never placing a disk on a smaller disk.

This problem has well-known matching upper and lower bounds of $2^n - 1$, where n is the number of disks [Ball & Coxeter '74]. The algorithm for the upper bound was the subject a number of papers that explored the amount of storage necessary for the solution. As it turns out, there is a simple iterative solution, so the storage required is constant [Hayes '77]. Refer to Figure 1 for the (space-wasting) recursive algorithm.

The Multi-Peg Towers of Hanoi Game

The multi-peg version of the game permits the use of k work pegs instead of the customary one. The classic game has $k+2$ pegs: the *source* peg, the *destination* peg, and one *work* peg. Here we have $k+2$ pegs, the source, the destination, and k work pegs. The problem is basically the same, to move the

```
MoveTower(n, s, d, w)
int n; /* number of disks to move */
peg s, d, w;   /* source, destination and work peg */
{
    if (n == 0) return;
    else if (n == 1)
        MoveDisk(s, d);
    else
    {
        MoveTower(n - 1, s, w, d);
        MoveDisk(s, d);
        MoveTower(n - 1, w, d, s);
    }
}
```

Fig. 1. Algoritnm 1, the classic recursive "three-peg" algorithm. MoveDisk(s, d) moves a single disk from peg s to peg d.

perfectly ordered tower of disks from the source to the destination one disk at a time, using the work pegs as needed, and never placing a disk on a smaller one. Throughout the remainder of this paper, $T(n, k)$ will represent the least amount of time required to move n disks from the source to the destination using k work pegs.

Proposition 1: For any $n, k > 0$, the time $T(n, k)$ required to move n disks from the source to the destination is at least $2n - 1$.

Proof:

The top $n - 1$ disks must be removed from the bottom disk before it may be moved, requiring at least $n - 1$ moves. The bottom disk may then be moved to the destination, requiring one move. At least $n - 1$ more moves are then required to place the smaller disks on the bottom disk, for a total of $2n - 1$.

⊠

This is the lower bound to move n disks in a particular order to another peg, in the same order, *regardless* of what ordering constraints exist, if only one disk may be moved at a time and the moves treat the pegs like stacks

(addition and removal only from the top of the peg). This bound may be achieved if there are sufficient work pegs.

Proposition 2: For any $n > 0$, if $k \geq n-1$ then

$$T(n, k) = T(n, n-1) = 2n-1.$$

Proof:

It is easy to see that the $n-1$ smaller disks may be spread out across the $n-1$ work pegs in time exactly $n-1$. The total time is then no more than

$$T(n, n-1) \leq n-1+1+n-1 = 2n-1.$$

Since this matches the lower bound, it is clearly the exact time. ☒

This linear upper bound when there are plenty of work pegs and the exponential lower bound when there is one work peg demand investigation into the behavior when there is an intermediate number of work pegs. This is easier said than done, however. The lower bound when there is but one work peg is relatively easy to establish because the extremely limited number of work pegs restricts what can possibly be done to continue to make progress. With additional pegs a lower bound has not been found, although it appears certain to be exponential if the number of pegs is constant.

The difficulty posed by analysis of games with additional pegs is that the intermediate configurations are not as constrained; with only one work peg, you must build a complete stack on the only available peg in order to free the next disk. With more pegs, you have the the choice of building lots of little towers quickly or a few big ones as intermediate configurations. The relevant tradeoffs appear to be :

(A) the time to build the (intermediate) tower(s)

(B) its (their) size

(C) the number of towers to build

If we choose the strategy of building one "working stack" at a time, then we retain all but one of the working pegs free for future use. By using the spread and stack algorithm (see Figure 2) as a subroutine, we can treat substacks of

```
Spread&Stack(n, s, d, W)
int n; /* number of disks to move */
peg s, d;  /* source and destination peg */
list of pegs W;  /* work pegs */
{
    if (n==0) return;
    else if (n==1)
        MoveDisk(s, d);
    else if (n>card(W)+1)
        halt;  /* not enough work pegs */
    else
    {
        wp = first(W);
        for (i=1; i<n; i++)
        {
            MoveDisk(s, wp);
            wp = next(wp);
        }
        MoveDisk(s, d);
        wp = first(W);
        for (i=1; i<n; i++)
        {
            MoveDisk(wp, d);
            wp = next(wp);
        }
    }
}
```

Fig. 2. Algorithm 2, "spread and stack". MoveDisk(s, d) moves a single disk from peg s to peg d; card(W) is the size of W, or the number of work pegs; next(wp) returns the next peg in W after wp.

size k as "superdisks", and use the standard (three-peg) algorithm on these. We may do this because there will always be $k-1$ work pegs available to move the superdisks (the first superdisk can be of size $k+1$, but for simplicity's sake, we will ignore this.) If we have $n = m \times k$ disks, then we require $2^m - 1$ moves of the superdisks, each requiring $2k-1$ moves, or a total of

$$T(n,k) = T(n,n/m) \leq (2k-1)(2^m-1)$$
$$= (2k-1)(2^{n/k}-1) = O(k2^{n/k})$$

when $k = n/m = O(n)$.

Proposition 3 : If the number of works pegs $k \geq n/f(n)$, then the time required to move n disks from the source to the destination is no more than

$$T(n, n/f(n)) \leq (2n/f(n)-1)(2^{f(n)}-1) = O((n/f(n))2^{f(n)}).$$

Proof:

Given above. ☒

We achieve this same bound if we use the strategy of building k substacks of size n/k using the three-peg algorithm. Clearly the failure to use the extra work pegs whenever possible wastes some of their potential. Still, this crude upper bound gives us some interesting results when we plug in certain functions for k. All three of these corollaries are direct applications of proposition 2.

Corollary 4 : If $k = \log n$, then

$$T(n, \log n) = O(2^n(\log n)/n).$$

Corollary 5 : If $k = n^a$, where a is a constant, then

$$T(n, n^a) = O(n^a 2^{n^{1-a}}).$$

Corollary 6 : If $k = n/\log n$, then

$$T(n, n/\log n) = O(n^2/\log n).$$

Another approach to solving the problem of space (the number of work pegs) and time tradeoffs is to determine what space is necessary for a linear time solution. Clearly, $k = n-1$ works pegs suffice. Here, each disk moves at most twice. What happens if we constrain each disk to move at most four times? (Note that three moves is too restrictive, using the observation that if we move a disk onto a larger disk on a work peg, then we will have to move it at least twice more, once to free the larger disk under it and once to put it into

```
FourMove(n, s, d, W)
int n; /* number of disks to move */
peg s, d;  /* source and destination peg */
list of pegs W;  /* work pegs */
{
    if (n==0) return;
    else if (n==1)
        MoveDisk(s, d);
    else if ((sqrt(8n+1)-3)/2>card(W))
        halt;  /* not enough work pegs */
    else
    {
        k=card(W);
        wp = first(W);
        WL=W-[wp];
        for (i=1; i<k; i++)
        {       /* must test for end conditions here */
            Spread&Stack(k+2-i, s, wp, WL+[d]);
            wp = next(wp);
            WL=WL-[wp];
        }
        MoveDisk(s, d);
        wp = first(W);
        WL=W-[wp];
        for (i=1; i<k; i++)
        {       /* must test for end conditions here */
            Spread&Stack(k+2-i, s, wp, WL+[d]);
            wp = next(wp);
            WL=WL-[wp];
        }
    }
}
```

Fig. 3. Algorithm 3, "FourMove". No disk moves more than four times. Subroutines appear in Figure 2.

place in the final stack, for a total of at least four moves.) The best we can do is to build towers using the spread and stack algorithm (using at most two moves apiece), then move the bottom disk to the destination peg, then move the towers in reverse order using spread and stack to the destination peg. This FourMove algorithm is shown in Figure 3. How many disks can we move in this manner?

Let $N(4, k)$ be the maximum number of disks we can move using at most four moves per disk and k work pegs. We can build k towers of size $k+1$, k, $k-1$, ..., 2 (each tower built decrements the number of work pegs available until only the destination peg is left) and then move the bottom disk to the destination, for a total of

$$N(4,k) = \sum_{i=1}^{k+1} i = \frac{(k+1)(k+2)}{2} = \frac{k^2+3k+2}{2}.$$

The total time $T_4(N(4, k), k)$ taken is

$$T_4 N(4,k),(k) = 2\sum_{i=1}^{k+1}(2i-1)-1 = 4\sum_{i=1}^{k+1} i - 2\sum_{i=1}^{k+1} 1 - 1 = 2k(k+2)+1$$

By solving for k in terms of n, we can determine that if

$$k \geq \frac{\sqrt{8n+1}\ -3}{2}$$

then

$$T(n,k) = T(n, \frac{\sqrt{8n+1}\ -3}{2}) = 4n - \sqrt{8n+1}\ = O(n).$$

Continuing in this line of reasoning, let us see what can be done by restricting each disk to move no more than eight time. Using the FourMove scheme to move as much as possible to each of the k work pegs, we may move $(k+1)(k+2)/2$ disks to the first peg, $k(k+1)/2$ to the second work peg, ..., 3 disks to the k^{th} peg. The bottom disk may be moved to the destination peg, and the process reversed. The number of disks we have moved is then

$$N(8,k) = 1 + \sum_{i=1}^{k} (i+1)(i+2)/2 = \frac{k^3 + 6k^2 + 11k + 6}{6}$$

with the time required

$$T_8(N(8,k),k) = 1 + 2\sum_{i=1}^{k}[2(i+1)^2 - 1] = \frac{2k^3 + 9k^2 + 10k + 3}{3}$$

Thus with fewer than $k=(n/6)^{1/3}$ pegs we can move n disks in time less than $8n$.

The following lemma will be used to establish a general upper bound of the sort derived above.

Lemma 7: For integers $n \geq 1, c \geq 0$,

$$\sum_{i=1}^{n} i^c \geq \frac{n^{c+1}}{(c+1)!}.$$

Proof:

This may be proven by observing that it is true for $n=1$, then assuming it is true for $n<N+1$. In that case,

$$\sum_{i=1}^{N+1} i^c = (N+1)^c + \sum_{i=1}^{N} i^c \geq \frac{N^{c+1}}{(c+1)} + (N+1)^c = \frac{N^{c+1}+(c+1)(N+1)^c}{(c+1)}$$

but

$$\frac{N^{c+1}+(c+1)(N+1)^c}{(c+1)} \geq \frac{(N+1)^{c+1}}{(c+1)}$$

since

$$(c+1)\binom{c}{j} \geq \binom{c+1}{j+1},$$

because

$$(c+1)\binom{c}{j} = (c+1)\frac{c!}{j!(c-j)!} = \frac{(c+1)!}{j!(c-j)!} > \frac{(c+1)!}{(j+1)!(c-j)!} = \binom{c+1}{j+1}.$$

Furthermore, equality only holds when $c=0$. ☒

We use this lemma to prove the following theorem.

Theorem 8: If $k = (c!n)^{1/c} \approx (c/e)n^{1/c}$ for some constant c, where e is the base of the natural logarithm, then $T(n, k) < 2cn$.

Proof:

We induct using the methods given above. The base case of $c=1$ is trivially true.

Suppose that the theorem holds for $c<C$. For $T(n, k)<2Cn$ moves, we (too sternly) insist that no disk make more than $2c$ moves (and know that at least one disk will make less than that, e.g.: the bottom disk, which will make exactly one move.) This can be enforced by using the strategy of building k

towers using no more than 2^{C-1} moves per disk, then moving the bottom disk, then reversing the tower building process to stack those disks on the bottom one. Clearly, this uses no more than 2^C moves per disk.

As before, we will fix k and c, then determine the maximum number of disks that can be moved under this strategy. Let $N(c, k)$ be the maximum number of disks that can be moved using no more than 2^c moves per disk.

Claim: $N(c, k) > k^c/c!$

Proof of Claim: It has already been shown for any k and for $c=2$. Suppose it is true for any k and for $c<C$. Then by using k work pegs to build the first tower, $k-1$ work pegs to build the second and so on, we can build k towers of a combined size

$$N(C, k) > \sum_{i=1}^{k} N(C-1, i) > \sum_{i=1}^{k} \frac{i^{(C-1)}}{(C-1)!},$$

$$= \frac{1}{(C-1)!} \sum_{i=1}^{k} i^{(C-1)} \geq \frac{1}{(C-1)!} \times \frac{k^C}{C} = \frac{k^C}{C!}.$$

This follows from lemma 1 above. Since $N(C, k) > k^C/C!$, this means that for any number of disks $n \leq N(C, k)$, we may move them in $2^C n$ moves. In other words,

$$k^C/C! < n \leq N(C, k), \text{ so}$$

$$k^C < C!n, \text{ or}$$

$$k < (C!n)^{1/C}.$$

Certainly, having extra work pegs cannot do us harm, so if we can get away with fewer than $(C!n)^{1/C}$ work pegs, we can do at least as well with more. ☒

Conjecture : For any average number of moves per disk 2^m (i.e.: linear constant), and any fixed number of work pegs k, there is some number $M(m, k)$

of disks such that at least $2^m \times M(m, k)$ moves are required to move $M(m, k)$ disks from one peg to another peg.

Peter Gacs, crediting others with similar but unpublished discoveries, derived the following upper bounds with some specific applications.

Theorem 9 (Gacs) : Let $m \geq k$ be a number such that $n \leq \binom{m}{k}$. Then

$$T(n, k) \leq 2^{m-k+1} min\{\binom{m}{k-1}, \binom{m}{k}\}.$$

Corollary 10 (Gacs) : If k is small with respect to n, so $m = \lceil kn^{1/k} \rceil + 1$, then

$$T(n, k) \leq n 2^{kn^{1/k}}.$$

Corollary 11 (Gacs) : If k is logarithmic with respect to n then $T(n, k)$ becomes polynomial in n. Further, if $\varepsilon_k = log\ k/k$, and

$$k = \lceil (\varepsilon_k + 1/2) log\ n \rceil,$$

then

$$T(n, k) \leq 2n^{1.5+\varepsilon_k}.$$

As a measure of the tightness of the upper bound, if $k = n-1$ and $m = n$, the inequality of the theorem gives $T(n, n-1) \leq 4n$.

In order to prove the theorem, the following lemma is used.

Lemma 12 (Gacs) : For any positive integers, n_0, n_1 with $n = n_0 + n_1$ we have

$$T(n, k) \leq T(n_0, k-1) + 2T(n_1, k).$$

Proof : (Gacs)

By moving n_1 disks to one peg using k work pegs, then moving n_0 disks to the destination peg using $k-1$ work pegs, then moving the n_1 disks to the destination peg using k work pegs, we obtain the desired inequality.

☒

Proof of the Theorem : (Gacs)

We prove it for $n = \binom{m}{k}$. The estimates from the min expression are

proven separately, by induction on m, starting with $m=k$. At $m=k$, we have $n=1$, so any positive integer upper bound is true.

For $k=1$, we have $m=n$, so $T(n, k)=2^n-1$, so the inequality is true.

Suppose that the relation holds for $m-1$ and that $k>1$. Using the identity

$$\binom{m}{l} = \binom{m-1}{l-1} + \binom{m-1}{l},$$

define

$$n_0 = \binom{m-1}{k-1}, \text{ and } n_1 = \binom{m-1}{k}.$$

Then by the lemma, the inductive assumption, the first corollary and the identity, we have

$$T(n, k) \le T(n_0, k-1) + 2T(n_1, k),$$

$$T(n, k) \le 2^{m-k+1}\binom{m-1}{k-1-i} + 2\times 2^{m-k}\binom{m-1}{k-i} = 2^{m-k+1}\binom{m}{k-i}.$$

☒

The theorem and its corollaries give better results than the naive approach of proposition 2 for non-linear functions, but they are better for $k=n/c$. The upper bound when $k=(c/e)n^{1/c}$ is also better. But what about lower bounds? Their difficulty has already been alluded to, but what can we really say about them? First we must define a new function, $F(n, t, k)$.

Definition: Let $F(n, t, k)$ be the number of moves required to transfer n disks from one peg to t target pegs using k additional work pegs.

It is easy to see that $T(n, k) = F(n, 1, k)$. Also obvious are the following inequalities.

(A) $F(n,t,k) \le F(n,t-1,k+1)$

(B) $F(n,t,k) \le F(n,t,k-1)$

(C) $F(n,t,k) \le min_{i,j}\{F(n-i,t-j,k+j)+F(i,j,k)\}$

(D) $F(n,t,k) > F(n-1,t,k)$

The first of these can be phrased, "It can't hurt to let a work peg be a target peg." The second can be phrased, "It can't hurt to have an additional work peg." The third is simply a statement that we can do at least as well as the obvious algorithm of moving some of the disks onto some of the target pegs, then moving the rest of the disks onto the rest of the target pegs. The last is simply, "It is strictly worse if there are more disks to move." Really, the inequality (A) should be strict, but this requires a more subtle proof. It should be pointed out that adding a disk and a target peg, or even a disk and a work peg, do *not* make things worse in general (it appears that in some cases that the time required actually decreases, witness $F(7, 1, 2)$ and $F(8, 1, 3)$ for example.)

We can use these to show some interesting equalities.

Theorem 13: The following equalities all hold.

(1) $F(n, n, 0) = n$.

(2) $F(n, 1, n-1) = 2n-1$.

(3) $F(n, 1, 1) = T(n, 1) = 2^n-1$.

(4) $F(n, t, n-t) = 2n-t$, where $t>0$.

(5) $F(n, 1, k) = 1+2F(n-1, k, 1) = T(n, k)$.

(6) $F(n, 2, 0) = F(n-1, 1, 1)+F(1, 1, 0) = 2^n-1$.

(7) $F(n, t, 0) = 1+F(n-1, t-1, 1)$.

(8) $F(n, t, k) = 2n-t$ if $t>1$ and $t+k>n/2$.

(9) $F(n, 1, k) = 2(2n-k-2)+1$, provided $k+1 \geq n/2$.

Proof:

(1) $F(n, n, 0) = n$.

Proof: obvious. ☒

(2) $F(n, 1, n-1) = 2n-1$.

Proof: Shown above. ☒

(3) $F(n,1,1) = T(n,1) = 2^n - 1$.

Proof: Well known and easy. ☒

(4) $F(n,t,n-t) = 2n-t$, where $t>0$.

Proof: True for all n if $t=1$ or $t=n$. If $t=n$, then

$$F(n,t,n-t)=F(n,n,0)=n=2n-t,$$

while if $t=1$, then

$$F(n,t,n-t)=F(n,1,n-1)=2n-1=2n-t.$$

By the simple algorithm of spreading the n disks out over the n target and work pegs, then moving the $n-t$ smallest disks onto a target peg (in largest to smallest order), we have

$$F(n,t,n-t) \le F(n,n,0)+F(n-t,n-t,0) = n+(n-t)$$

$$F(n,t,n-t) \le 2n-t.$$

We may spread n disks over $t>1$ target pegs, reserving the smallest disk by itself on one peg in $2n-t$ moves. This can be seen by the application of (C), letting $i=t-1$.

$$\begin{aligned} F(n,t,n-t) &\le F(n-i,t-i,n-t+i)+F(i,i,n-t) \\ &\le F(n-(t-1),1,n-1)+F(t-1,t-1,n-t) \\ &= [2(n-t+1)-1]+(t-1) = 2n-t, \end{aligned}$$

The last step uses (2) above.

Therefore, for $t-1$ target pegs, we can use this method to move the n disks onto t pegs with the smallest disk on a work peg. Then we move the smallest disk from the work peg onto one of the the target pegs, so

$$F(n,t-1,n-t+1) \le F(n,t,n-t)+1.$$

This gives us

$$F(n,t,n-t) \ge F(n,t-1,n-t+1)-1.$$

Assuming the induction hypothesis for $t'<t$, we have

$F(n, t, n-t) \geq 2n-(t-1)-1 = 2n-t.$

Since $2n-t \geq F(n, t, n-t) \geq 2n-t$, the hypothesis holds for all t. ☒

(5) $F(n, 1, k) = 1+2F(n-1, k, 1) = T(n, k)$.

Proof: For the ordering constraint to be observed, we must first remove the top $n-1$ disks off the bottom disk and onto the available pegs, leaving one free to accept the bottom disk, then move the bottom disk. The best we can do to stack the other disks on top of the bottom disk is to reverse the moves used to spread them over the k pegs, with appropriate substitution of pegs. Otherwise, we would have a faster method of spreading $n-1$ disks over k pegs. The total time is then $F(n-1, k, 1)+1+F(n-1, k, 1)$, as desired. ☒

(6) $F(n, 2, 0) = F(n-1, 1, 1)+F(1, 1, 0) = 2^{n-1}$.

Proof: The top $n-1$ disks must be removed from the bottom disk onto one target peg, using the other as a work peg. Only then may the bottom disk may be moved. ☒

(7) $F(n, t, 0) = 1+F(n-1, t-1, 1)$.

Proof: Similar to the proof of (6). ☒

(8) $F(n, t, k) = 2n-t$ if $t>1$ and $t+k>n/2$.

Proof: We use equality (4) and inequality (B),

$2n-t = F(n, t, n-t) \leq F(n, t, k)$.

Now we use inequality (C) and equality (4),

$$F(n, t, k) \leq F(t+k, 1, t+k-1)+F(n-(t+k), t-1, k)$$
$$= [2(t+k)-1]+[2(n-(t+k))-(t-1)] = 2n-t.$$

Equality (4) may be applied to the second term because

$n-(t+k) \leq (t-1)+k = t+k-1$,

which is true if and only if

$n \leq 2(t+k)-1$,

which is equivalent to our requirement that

$n/2 < t+k$. ☒

(9) $F(n, 1, k) = 2(2n-k-2)+1$, provided $k+1 \geq n/2$.

Proof: Using (5) we have

$F(n, 1, k) = 2F(n-1, k, 1)+1$.

Since our requirement that $k+1 \geq n/2$ implies that $k+1 > (n-1)/2$, we may apply (8) to the right hand side to obtain

$F(n, 1, k) = 2[2(n-1)-k]+1$,

as desired. ☒

To sum up these results, one can say that F is "well-behaved" and "small" as long as $t+k > n/2$. Outside of this range, it seems to really blow up. Some remarkable things can be seen to occur. First, note that even when $t+k<n$, in some cases it does not help to add an extra work peg (see (4) and (8)). Second, there is a general lower bound of $4n-2k-3$ on $T(n, k)$ given by (9).

Open Questions

It would be highly desirable to prove that inequality (C) is actually equality, that inequality (A) is strict in certain ranges and that inequality (D) can be strengthened. Since (B) cannot be improved upon, the list should really be

(A') $F(n, t, k) < F(n, t-1, k+1)$ when $t+k < n/2-1$ or $k=1$

(B) $F(n, t, k) \leq F(n, t, k-1)$

(C') $F(n, t, k) = min_{i,j}\{F(n-i, t-j, k+j)+F(i, j, k)\}$

(D') $F(n, t, k) > F(n-1, t, k)+1$ when $t<n$.

The last of these has the condition that $t<n$ since when $t=n$, all that is required is to move the additional disk onto the empty target peg.

A general form for $F(n, t, k)$ is still sought, or at least bounds better than those given here. In particular, better understanding of the behavior of $F(n, t, k)$ when k and t are small is necessary. Finding tight general bounds for $T(n, 1, k) = T(n, k)$ is the most interesting open question. A seemingly obvious but tricky conjecture will be our parting note.

Conjecture : For k constant, $T(n, k)$ is exponential.

References

Aho, A., J. Hopcroft and J. Ullman ['74], The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

Atkinson, M. D. ['81], "The Cyclic Towers of Hanoi," *Information Processing Letters* 13:3, 118-119, December 1981.

Ball, W. W. Rouse ['92], Mathematical Recreations and Essays, McMillan, London, 1892.

Ball, W. W. Rouse and H. S. M. Coxeter ['74], Mathematical Recreations and Essays, 12th edition, University of Toronto Press, Toronto, 1974.

Buneman, Peter and Leon Levy ['80], "The Towers of Hanoi Problem," *Information Processing Letters* 10, 243-244, July 1980.

Engelfriet, Joost ['81], "The Trees of Hanoi," Twente University of Technology Technical Report, 1981.

Er, M. C. [82], "A Representation Approach to the Tower of Hanoi Problem," *The Computer Journal* 25:4, 442-446, 1982.

Er, M. C. [84a], "The Colour Towers of Hanoi: A Generalization," *The Computer Journal* 27:1, 80-82, February 1984.

Er, M. C. [84b], "The Cyclic Towers of Hanoi: A Representation Approach," *The Computer Journal* 27:2, 171-175, May 1984.

Er, M. C. [84c], "The Generalized Colour Towers of Hanoi: An Iterative Algorithm," *The Computer Journal* 27:3, 278-282, August 1984.

Gacs, Peter ['86], "The Generalized Hanoi Towers Problem," *manuscript*, Computer Science Department, Boston University, May 1986.

Gardner, Martin ['72], "Mathematical games: the Curious Properties of the Gray Code and How It Can Be Used to Solve Puzzles," *Scientific American*, 106-109, August 1977.

Hayes, Patrick J. ['77], "A Note on the Towers of Hanoi Problem," *The Computer Journal* 20:3, 282-285, August 1977.

Jurgensen, Helmut and Derick Wood ['83], "The Multiway Trees of Hanoi," *International Journal of Computer Mathematics* 14:2, 137-153, 1983.

Partsch, H. and P. Pepper ['76], "A Family of Rules for Recursion Removal," *Information Processing Letters* 5:6, 174-177, 1976.

Walsh, T. R. ['82], "The Towers of Hanoi Revisited: Moving the Rings by Counting the Moves," *Information Processing Letters* 15:2, 64-67, September 1982.

Walsh, T. R. ['83], "Iteration Strikes Back - At the Cyclic Towers of Hanoi," *Information Processing Letters* 16:2, 91-93, February 1983.

Wirth, N. ['76], Algorithms + Data-Structures = Programs, Prentice-Hall, Inc. Englewood Cliffs, NJ, 1976.

Wood, Derick ['81], "The Towers of Hanoi or Brahma Revisited," *Journal of Recreational Mathematics* 14, 17-24, 1981.

Appendix

Values for the function $F(n, t, k)$ for some small values of n, $n \le t+k \le 2$. Only the parameters are given, and the columns roughly correspond to constant $t+k$. The $\le$ sign is used where Theorem 13 does not cover and inequality C is used.

$(2,2,0)=2$ $(2,1,1)=3$

$(3,3,0)=3$ $(3,2,1)=4$ $(3,2,0)=4$
$(3,1,2)=5$ $(3,1,1)=7$

$(4,4,0)=4$ $(4,3,1)=5=(4,3,0)=5$
$(4,2,2)=6=(4,2,1)=6$ $(4,2,0)=8$
$(4,1,3)=7$ $(4,1,2)=9$ $(4,1,1)=15$

$(5,5,0)=5$ $(5,4,1)=6=(5,4,0)=6$
$(5,3,2)= \ldots =(5,3,0)=7$
$(5,2,3)= \ldots =(5,2,1)=8$ $(5,2,0)=16$
$(5,1,4)=9$ $(5,1,3)=11$ $(5,1,2)=13$ $(5,1,1)=31$

$(6,6,0)=6$ $(6,5,1)=7=(6,5,0)=7$
$(6,4,2)= \ldots =(6,4,0)=8$
$(6,3,3)= \ldots =(6,3,0)=9$
$(6,2,4)= \ldots =(6,2,2)=10$ $(6,2,1) \le 12$
$(6,1,5)=11$ $(6,1,4)=13$ $(6,1,3)=15$ $(6,1,2)=17$ $(6,1,1)=63$

$(7,7,0)=7$ $(7,6,1)=8=(7,6,0)=8$
$(7,5,2)= \ldots =(7,5,0)=9$
$(7,4,3)= \ldots =(7,4,0)=10$
$(7,3,4)= \ldots =(7,3,1)=11$ $(7,3,0) \le 13$
$(7,2,5)= \ldots =(7,2,2)=12$ $(7,2,1) \le 16$ $(7,2,0)=64$
$(7,1,6)=13$ $(7,1,5)=15$ $(7,1,4)=17$ $(7,1,3)=19$ $(7,1,2) \le 25$
$(7,1,1)=127$

$(8,8,0)=8$ $(8,7,1)=9=(8,7,0)=9$
$(8,6,2)= \ldots =(8,6,0)=10$
$(8,5,3)= \ldots =(8,5,0)=11$
$(8,4,4)= \ldots =(8,4,0)=12$
$(8,3,5)= \ldots =(8,3,1)=13$ $(8,3,0) \le 17$
$(8,2,6)= \ldots =(8,2,3)=14$ $(8,2,2) \le 16$ $(8,2,1) \le 20$ $(8,2,0)=128$
$(8,1,7)=15$ $(8,1,6)=17$ $(8,1,5)=19$ $(8,1,4)=21$ $(8,1,3)=23$
$(8,1,2) \le 33$ $(8,1,1)=255$

$(9,9,0)=9$ $(9,8,1)=10=(9,8,0)=10$
$(9,7,2)= \ldots =(9,7,0)=11$
$(9,6,3)= \ldots =(9,6,0)=12$
$(9,5,2)= \ldots =(9,5,0)=13$
$(9,4,5)= \ldots =(9,4,0)=14$
$(9,3,6)= \ldots =(9,3,1)=15$
$(9,2,7)= \ldots =(9,2,3)=16$ $(9,2,2) \le 20$ $(9,2,1) \le 24$ $(9,2,0)=256$
$(9,1,8)=17$ $(9,1,7)=19$ $(9,1,6)=21$ $(9,1,5)=23$ $(9,1,4)=25$
$(9,1,3)=27$ $(9,1,2) \le 41$ $(9,1,1)=511$

END

9-87

DTIC